

Логика работы.

На сервере создается таблица и заполняется данными. Клиент подключается и через класс репликации получает все объекты которые есть на сервере. Объекты приходят без данных.

На клиенте получаем репликацию таблицы на ее основе создает класс отображения таблицы. Для отображения таблица запрашивает данные в реплики таблицы. Таблица смотрит эти данные если их нет, то она запрашивает через модуль репликации с сервера. Посылается сообщение `MsgReplicationNeedData` с указанием ID объекта

На данный момент реализовано получения реплики таблицы на клиент и создание на ее основе объекта отображения данных.

Ссылка на репозиторий:

https://github.com/galikt/test_work.git

Почта для связи:

vgerasimchuk90@gmail.com

Описание классов

Базовый класс для сериализации десериализации. Использует `TWBuffer`.

```
class TWSerialization
    virtual void Serialize(std::ostream& stream);
    virtual void Deserialize(std::istream& stream);
```

Класс предоставляет массив `char` для хранения и передачи по сети. Запись и чтение в буфер производится через потоковый ввод

```
class TWBuffer : public std::streambuf
    char* GetData();
    size_t GetSize();
    size_t GetCapacity();
    std::istream GetStream();
```

Базовый класс. Реализует отправку и получение сообщений через класс `TWMessenger`.

```
class TWObjectBase : public TWSerialization
    void SendMsg(std::unique_ptr<TWMessage>&& msg);
    virtual void ProcessingMsg(std::unique_ptr<TWMessage>&& msg);
```

Указывает что объект не оригинальный. На сервере есть оригинал объекта.

```
bool Replica{ false };
```

Класс хранит строку таблицы в виде вектора

```
class TWRow : public TWObjectBase
    std::vector<char> Data;
```

Базовый класс столбец. Содержит название столбца и смещение в массиве для извлечения данных из `TWRow`

```
template<typename DataType>
class TWColumn : public TWColumnBase
    virtual size_t GetDataSize();
    virtual std::string GetData(std::shared_ptr<TWRow>& row);
```

```
DataType Data;
```

Класс таблицы. Вставка столбцов формирует структуру объекта TWRow.

```
class TWTable : public TWObjectBase
    void InsertColumn(std::unique_ptr<TWColumnBase> column);
    void InsertRow(std::shared_ptr<TWRow> row);
```

Отсортированная таблица по id TWRow

```
std::map<uint32_t, std::shared_ptr<TWRow>> RowMap;
```

Отсортированная таблица по колонке

```
std::map<uint32_t, std::shared_ptr<TWRow>> SortedMap;
```

Класс для отображения таблицы

```
class TWTableView : public TWObjectBase
    void SetTable(std::shared_ptr<TWObjectBase> obj);
```

Класс создает объекты по типу класса

```
class TWMetaObject
    static std::shared_ptr<TWObjectBase> CreateObject(ObjectType type);
    static std::unique_ptr<TWMessage> CreateMessage(ObjectType type);
    static std::unique_ptr<TWColumnBase> CreateColumn(ObjectType type);
```

Класс хранит объекты и предоставляет к ним доступ

```
class TWObjectContainer
    virtual void RegisterObject(std::shared_ptr<TWObjectBase> obj);
    std::shared_ptr<TWObjectBase> FindObject(ObjectType type);
```

Базовый класс для клиента и сервера.

```
class TWGate
    virtual void TransmitBuffer(uint32_t id, std::unique_ptr<TWBuffer>&& buf);
    virtual void ReceiveBuffer(uint32_t id, std::unique_ptr<TWBuffer>&& buf);
    virtual void Connected(uint32_t id);
```

Класс сервер. Передает и принимает TWBuffer клиентам

```
class TWGateServer : virtual public TWGate
    virtual void DeInit() override;
    virtual void TransmitBuffer(uint32_t id, std::unique_ptr<TWBuffer>&& buf) override;
    std::map<uint32_t, TWClientContext> ClientMap;
```

Класс клиент.

```
class TWGateClient : virtual public TWGate
    virtual void TransmitBuffer(uint32_t id, std::unique_ptr<TWBuffer>&& buf) override;
```

Класс реализует передачу сообщений между TWObjectBase. Получает TWBuffer от TWGate.

Десериализует буфер и добавляет сообщения в пул сообщений.

```
class TWMessenger : virtual public TWGate, public TWObjectContainer
    void SendMsg(std::unique_ptr<TWMessage>&& msg);
    virtual void ReceiveBuffer(uint32_t id, std::unique_ptr<TWBuffer>&& buf) override;
    std::list<std::unique_ptr<TWMessage>> MessageList;
```

Базовый класс для репликации

```
class TWReplication : public TWMessenger
    virtual void ReceiveBuffer(uint32_t id, std::unique_ptr<TWBuffer>&& buf) override;
```

Класс занимается передачей объектов на клиент

```
class TWReplicationServer : public TWReplication
    virtual void ProcessingReplication() override;
    virtual void Connected(uint32_t id) override;
```

Получает буфер от TWGate. Если сообщение для репликации, то сериализует его и добавляет объект репликации в список объектов. Иначе передает буфер в TWMessenger

```
class TWReplicationClient : public TWReplication
    virtual void ProcessingReplication() override;
    virtual void Connected(uint32_t id) override;
    virtual void RegisterObject(std::shared_ptr<TWObjectBase> obj) override;
```

Класс выступает объединением функционала. Базовый класс для создания клиента

```
class TWPlatformClientBase : public TWPlatform, public TWGateClient, public
TWReplicationClient
```

Класс выступает объединением функционала. Базовый класс для создания сервера

```
class TWPlatformServerBase : public TWPlatform, public TWGateServer, public
TWReplicationServer
```

В этом классе должна реализовываться пользовательская логика клиента

```
class TWClient : public TWPlatformClientBase
    virtual void Run() override;
```

В этом классе должна реализовываться пользовательская логика сервера

```
class TWServer : public TWPlatformServerBase
    virtual void Run() override;
```

Класс который работает с таблицей. Модифицирует ее раз в секунду.

```
class TWMonkey
```